Amendments to the Claims

1. (currently amended) A method to analyze a computer program that includes a plurality of executable blocks of code, the method comprising:

receiving a block of code to a code cache;

using a code block frequency counter for tracking each time said block of code is executed on said code cache, wherein additional code for incrementing said code block frequency counter is dynamically added to said block of code as said computer program is executed;

maintaining a counter cache for storing each said code block frequency counter of said block of code while said block of code is stored on said code cache, wherein said counter cache is distinct from said code cache, such that when said block of code is executed said code block frequency counter is readily available for tracking said execution; ~~and~~

maintaining a storage area for storing each said code block frequency counter of said block of code previously executed on said code cache, said storage area distinct from said counter cache and said code cache, said code block frequency counter being stored in said storage area after said block of code is evicted from said code cache, said block of code being evicted to make room for another block of code in said code cache, said code block frequency counter being stored for future access in a location which is distinct from said code cache and said counter cache; and

in the event that said block of code is again received into said code cache for execution subsequent to being evicted, copying said code block frequency counter from

said storage area to said counter cache to enable continuation of said tracking each time said block of code is executed on said code cache.

2. (previously presented) The method of Claim 1, further comprising the step of:

identifying when said code cache is full.

3. (canceled)

4. (previously presented) The method of Claim 2, further comprising:

determining which said code block frequency counter of said block of code stored on said counter cache is least recently executed;

evicting said least recently executed block of code, related to said code block frequency counter, from said code cache; and

copying said code block frequency counter of said least recently executed block of code from said counter cache to said storage area when said least recently executed block of code related to said code block frequency counter is evicted from said code cache.

5. (previously presented) The method of Claim 1, wherein said receiving a block of code to a code cache further comprises:

checking said storage area to determine if said block of code is being executed for other than the first time;

loading said code block frequency counter associated with said block of code being executed for other than the first time, from said storage area into said counter cache; and

updating said code block frequency counter associated with said block of code being executed for other than the first time.

6. (currently amended) A computer implemented system having a computer for analyzing a computer program that includes a plurality of blocks of code, comprising:

means for executing said computer program;

means for maintaining a code cache for storing at least one of a plurality of blocks of code derived from said computer program;

means for counting each time one of said plurality of blocks of code is executed, wherein additional code for incrementing said code counting means is dynamically added to said one of said plurality of blocks ~~block~~ of code as said computer program is executed;

means for maintaining a counter cache for storing said counting means of said plurality of blocks of code that are most recently executed, wherein said counter cache is distinct from said code cache, such that when said one of said plurality of blocks of code is executed said counting means is readily available for tracking said execution; and

means for maintaining a storage area for storing said counting means of said plurality of blocks of code that are not most recently executed, said storage area distinct from said counter cache and said code cache, said code counting means being stored

in said storage area after said block of code related to said counting means is evicted from said code cache, said block of code being evicted to make room for another block of code in said code cache, said counting means being stored for future access in a location which is distinct from said code cache and said counter cache, such that, subsequent to being evicted, if said block of code related to said counting means is again received into said code cache, said code counting means may be copied from said storage area to said counter cache to enable continuation of said tracking each time said block of code related to said counting means is executed within said code cache.

7. (previously presented) The system of Claim 6, further comprising:

means for identifying when said code cache is full.

8. (previously presented) The system of Claim 7, further comprising:

means for copying said counting means of said plurality of blocks of code from said code cache to said storage area when said code cache is full.

9. (previously presented) The system of Claim 8, wherein said identifying means further comprises:

means for determining which said counting means of said plurality of blocks of code in said code cache is least recently executed;

means for evicting said least recently executed block of code, related to said counter, from said code cache; and

means for copying said counting means, related to said least recently executed block of code, from said code cache to said storage area when said code cache is full.

10. (previously presented) The system of Claim 8, further comprising:

means for checking a code cache to determine if a block of code is being executed for other than the first time; and

means for loading said counting means associated with said block of code being executed for other than the first time, into said counter cache.

11. (currently amended) A computer ~~readable~~ storage medium having computer-readable program code embodied therein for causing a computer system to perform a method for analyzing a computer program that includes a plurality of executable blocks of code comprising:

receiving a block of code to a code cache;

utilizing a code block frequency counter for tracking each time said block of code is executed on said code cache, wherein additional code for incrementing said code block frequency counter is dynamically added to said block of code as said computer program is executed;

maintaining a counter cache for storing each said code block frequency counter of said block of code while said block of code is stored on said code cache, wherein said counter cache is distinct from said code cache, such that when said block of code is executed said code block frequency counter is readily available for tracking said execution; ~~and~~

maintaining a storage area for storing each said code block frequency counter of said block of code previously executed on said code cache, said storage area distinct from said code cache and said counter cache, said code block frequency counter being stored in said storage area after said block of code is evicted from said code cache, said block of code being evicted to make room for another block of code in said code cache, said code block frequency counter being stored for future access in a location which is distinct from said code cache and said counter cache; and

in the event that said block of code is again received into said code cache for execution subsequent to being evicted, copying said code block frequency counter from said storage area to said counter cache to enable continuation of said tracking each time said block of code is executed on said code cache.

12. (previously presented) The computer ~~readable~~ storage medium of Claim 11, further comprising:

identifying when said code cache is full.

13. (canceled)

14. (previously presented) The computer ~~readable~~ storage medium of Claim 12, further comprises:

determining which said code block frequency counter of said block of code in said counter cache is least recently executed;

evicting said least recently executed block of code, related to said code block frequency counter, from said code cache; and

copying said code block frequency counter of said least recently executed block of code from said counter cache to said storage area when said least recently executed block of code related to said code block frequency counter is evicted from said code cache.

15. (currently amended) The computer ~~readable~~ storage medium of Claim 12 ~~Claim 13~~, wherein said receiving a block of code to a code cache further comprises:

checking said storage area to determine if said block of code is being executed for other than the first time;

loading said code block frequency counter associated with said block of code being executed for other than the first time, from said storage area into said counter cache; and

updating said code block frequency counter associated with said block of code being executed for other than the first time.

16. (currently amended) A system for analyzing a computer program that includes a plurality of executable blocks of code, the system comprising:

a computer system having a computer configured for executing a block of code, said computer system comprising a cache memory;

a code block frequency counter that tracks each time a specific block of code is executed by a code cache, wherein additional code for incrementing said code block

frequency counter is dynamically added to said block of code as said computer program is executed;

a counter cache for storing said code block frequency counter of a specific block of code while said specific block of code is stored on said code cache, such that when said block of code is executed said code block frequency counter is readily available for tracking said execution, wherein said counter cache is distinct from said code cache; and

a storage area for storing said code block frequency counter of a specific block of code previously executed on said code cache, said storage area distinct from said counter cache and said code cache, said code block frequency counter being stored in said storage area after said specific block of code is evicted from said code cache to make room for another block of code in said code cache, said code block frequency counter being stored for future access in a location which is distinct from said code cache and said counter cache such that, in the event that said block of code is again received into said code cache for execution, subsequent to being evicted, said code block frequency counter may be copied from said storage area to said counter cache to enable continuation of said tracking each time said block of code is executed on said code cache.

17. (previously presented) The system of Claim 16, further comprising:

logic that identifies when said code cache is full.

18. (canceled)

19. (previously presented) The system of Claim 17, wherein said logic determines which said code block frequency counter of said specific block of code stored on said counter cache is least recently executed, evicting said least recently executed block of code related to said code block frequency counter from said code cache, and copies said code block frequency counter of said specific block of code from said counter cache to said storage area when said least recently executed specific block of code is evicted from said code cache.

20. (previously presented) The system of Claim 17, wherein said logic checks said storage area to determine if said specific block of code is being executed for other than the first time, and loads said code block frequency counter associated with said specific block of code being executed for other than the first time, from said storage area into said counter cache, and updating said code block frequency counter associated with said specific block of code being executed for other than the first time.